ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Informatik I D-ITET

Wintersemester 2006/07

Prof. François E. Cellier, Dirk Zimmer, Christoph Angerer, Irina Carabus, Daniel Keller, Mathias Payer, Philipp Bichsel, Patrick Bönzli, Theodor Mader, Robert Weiser

Supplementary Homework

Ausgabe: 20.12.2006
Abgabe: 22.01.2007

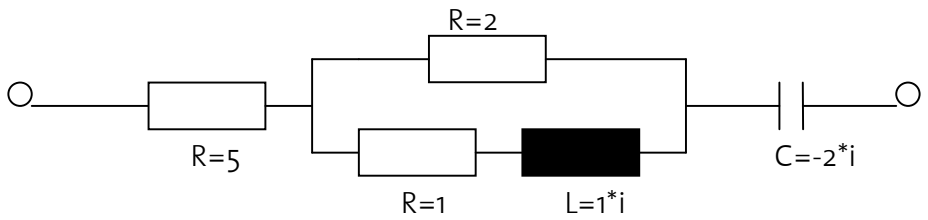
1. RLC circuit

Given a simple RLC circuit (and the values for each circuit component), write a C/C++ program that computes the impedance of that circuit. The impedance is like a resistance represented by a complex number. Within a circuit that is driven by a source of tension of a constant frequency (which is the case here), the impedance can express even inductances and capacitances.
Complex impedances can be seen like common resistances for computing equivalent elements. Aside from the fact that you have to compute the equivalent using complex numbers, the same rules apply.

Input:
The RLC circuit is given through an XML-like input file that may contain one or more of the following tags: "circuit", "parallel", "series", "resistance", "coil" and "condenser". "Circuit" is the root tag and appears only once. "parallel" and "series" tags specify the structure of the circuit elements. For every circuit element (resistance, coil or condenser) the value is given. Tags appear in format: <start_tag></end_tag>; for circuit elements, the values are given separated by quotations (Ex: <resistance>"1"</resistance>).

The impedance of a resistor is the resistance itself, represented by a real value. Capacitances and inductances are represented by pure imaginary numbers. In the case of a "coil", the number denotes the positive imaginary part whereas in the case of a "condenser" the number represents the negative imaginary part.

Example circuit:



Input file for example circuit:

```
<circuit>
     <series>
          <resistance>"5"</resistance>
          <parallel>
               <resistance>"2"</resistance>
               <series>
                    <resistance>"1"</resistance>
                    <coil>"1"</coil>
               </series>
          </parallel>
          <condenser>"2"</condenser>
     </series>
</circuit>
```

We make the assumption that the input file is well-formed:
- all open tags are closed at some point
- all tags close in the reverse order of their opening
- all values are contained in-between circuit elements tags
- all tags appear in-between "<" and "/>" characters
- all values appear in-between quotations

Output: A value representing the impedance of the equivalent circuit . For the example presented : 5.8-i*1.6.

Implementation issues and hints:
The XML-like input is given in a text file which is to be read and parsed. Parsing should search for strings delimited by "<","/>" or by quotations. All blanks are skipped. As computing the impedance of the equivalent circuit will require complex numbers, there should be a structure for holding complex numbers.

Example:

```
struct complex{
     float real;
     float imag;
};
```

Also basic methods for adding, dividing, substracting two complex numbers should be provided.

The easiest way for computing is to use a stack. While parsing the input file, the start tags and circuit elements values are put on the stack. When an end tag is encountered (Ex: "</coil>"), everything until the matching start tag in the stack is popped. Depending on the matching tag, the values for circuit elements are summed, substracted etc. and the result is pushed back on the stack.

2. Electricity sources plan

Suppose you are working in an electricity supply company. In order to make a plan for the next six months, you have to decide, which types of sources (coal, wind, water) are needed in producing electricity. In the process of decision, you have to take into account the customers' preferences, but in the same time to try minimizing the ecological impact, by choosing "cleaner" electricity sources.

Electricity types have prices per KWh depending on the ecological impact and how difficult it is to generate it. For example, nuclear power is cheap, but electricity produced from wind is more expensive.

Your company also keeps some statistics related to the electrical power consumption in the past years. Using this information, the company predicts for each type of electricity source, what is the production capacity (how many customers can be served). There is also a source of electricity, which is assumed to have an unlimited production capacity and this is of course the cheapest.

Every customer, in their wish to be more environmental-friendly, are willing to pay an extra amount of money per KWh over the cheapest price so that they would support the production of more ecological energy. A source of electricity is included in the final plan if the number of requests for that type is equal to the production capacity (the requests value is given by the number of the customers which are willing to pay the requested price per KWh for that source).

Given a set of electricity sources and a set of preferences from customers that request electricity from your company, write a simple C/C++ program which decides which types of sources are to be used for producing electricity in the next semester and, for each type of source, the percentage from the total production (we assume that, in average, every customer consumes the same amount of electricity).

Input: You are given two input files : **customers.dat** and **power.dat**.[1]

The first file contains information about the customers' preferences. On the first line, there is the number of customers. On the second line, for each customer, there is the amount of money (in MU = Money Units) per KWh they are willing to pay over the lowest price (separated by blanks) .

Example (customers.dat) :

```
6
7  2  6  3  6  4
```

[1] Please note The values are of course hypothetical, and were chosen for simplicity. The data is stored in text format.

The second file ("power.dat") contains the types of electricity sources with prices and availability. On the first line, there is the number of sources. On the following lines, for each type of electricity source, separated by blanks, there are : the name of the source, the price (in MU)/KWh and how many customers can be served with this type of power (availability).

Example (power.dat):

```
5
nuclear   10 1.000.000
gas       12 3
coal      14 3
water     16 2
wind      18 1
```

Observations:
For "nuclear", 1.000.000 capacity means that, using this source, any amount of electricity can be produced. The electricity sources are considered to appear in the file in the ascending order of their price per KWh. So that a price offered by a customer to qualify for a certain type of electricity source, it has to be greater or equal to the price per KWh for that source.

Output:

```
Water     2    33.33 %
Gas       3    50 %
Nuclear   1    16.66 %
```

Implementation issues and hints:
In the previous example, we add for each customer the extra amount they are willing to pay over the lowest price (which is for nuclear energy, as depicted from file **power.dat**). Then the list of prices the customers are willing to pay is the following:

17 12 16 13 16 14

The first customer is willing to pay 17 MU (money units) per KWh. This means that he/she can sustain *water* energy production so we increase the number of requests for *water* as a source of electricity:

Water requests : 1

The second customer increases the requests for *gas* with one as he/she is willing to pay 12 MU/KWh:

Water requests : 1
Gas requests : 1

The third customer increases the number of requests for *water* by 1 so the capacity for this type of source is reached (*water* will certainly be one of the electricity sources that will be used):

Water requests : 2
Gas requests : 1

The third customer increases the requests for *gas* by 1 (not for *coal* as he/she offers 13 MU < 14 MU which is the price per KWh for *coal*), so the number of requests for this sources becomes 2.

Water requests : 2
Gas requests : 2

The next customer offers 16 MU and that price would have qualified for *water*, but as the capacity for *water* is already exceeded, the request is pushed to the next source, which is *coal*:

Water requests : 2
Gas requests : 2
Coal requests : 1

The last customer offers 14 MU so the list of requests looks like this:

Water requests : 2
Gas requests : 2
Coal requests : 2

A source of electricity is used only if the requests reach the capacity, so *water* will be one of them. For *coal* we only have 2 requests, lower than 3 which is the capacity. Therefore these requests will be pushed to the next source, *gas*. For *gas* we have 2 requests plus 2 more coming from *coal*, but the capacity is 3, so we consider the capacity reached (*gas* is another source that will be added to the final plan) and push one request down to *nuclear*.

Therefore the assignment will be as follows:

Water	2	33.33 %
Gas	3	50 %
Nuclear	1	16.66 %